The arydshln package*

Hiroshi Nakashima
(Kyoto University)

2019/02/21

Abstract

This file gives LaTeX's `array` and `tabular` environments the capability to draw horizontal/vertical dash-lines.

Contents

*This file has version number v1.76, last revised 2019/02/21.

1 Introduction

In January 1993, Weimin Zhang kindly posted a style hvdashln written by the author, which draws horizontal/vertical dash-lines in LaTeX's `array` and `tabular` environments, to the news group comp.text.tex. The style, unfortunately, has a known problem that vertical lines are broken when an array contains tall rows.

In March of the year, Monty Hayes complained of this problem encouraging the author to make a new version arydshln to solve the problem. The new style also has new features, such as allowing '`:`' to specify a vertical dash-line in preamble, and `\cdashline` being a counterpart of `\cline`.

In March 1999, Sebastian Rahtz kindly invited the style, which had been improved following the bug report from Takahiro Kubota, to be included in TeX CTAN and also in the online catalogue compiled by Graham Williams. This invitation gave the style new users including Peter Ehrbar who wished to use it with array style in Standard LaTeX Tools Bundle and had trouble because these styles were incompatible with each other. Therefore, the style became compatible with array and got additional new features.

In February 2000, Zsuzsanna Nagy reported that arydshln is not compatible with colortab style to let the author work on the compatibility issue again.

In Feburary 2001, Craig Leech reported another compatibility problem with longtable. Although the author promised that the problem would be attacked some day, the issue had left long time[1] until three other complaints were made. Then the author attacked the problem hoping it is the last compatibility issue[2].

In May 2004, Klaus Dalinghaus found another incompatibility with colortbl. Although he was satisfied by a quick hack for cell painting, the author attacked a harder problem for line coloring to solve the problem[3].

2 Usage

2.1 Loading Package

The package is usable to both LaTeX 2_ε and LaTeX-2.09 users with their standard package loading declaration. If you use LaTeX 2_ε, simply do the following.

```
\usepackage{arydshln}
```

If you still love LaTeX-2.09, the following is what you have to do.

```
\documentstyle[..,arydshln,...]{⟨style⟩}
```

Only one caution given to users of array (v2.3m or later) and longtable (v4.10 or later) packages, included in Standard LaTeX Tools Bundle, and colortab and colortbl package is that arydshln has to be loaded *after* array, longtable, colortab and/or colortbl done. That is, the following is correct but reversing the order of `\usepackage` will cause some mysterious error.

[1] Two years and a half! Sorry Craig.
[2] But his hope was dashed as described below.
[3] Without dreaming it is the last compatibility issue.

```
\usepackage{array}     % and/or
\usepackage{longtable} % and/or
\usepackage{colortab}  % or
\usepackage{colortbl}
\usepackage{arydshln}
```

2.2 Basic Usage

array
tabular

You can simply use `array` or `tabular(*)` environments with standard preamble, such as `{r|c|ll}`, and standard commands `\\`, `\hline`, `\cline` and `\multicolumn`.

:

Drawing a vertical dash-line is quite simple. Use '`:`' in the preamble as the separator of columns separated by the dash-line, just like using '`|`' to draw a vertical solid-line. The *preamble* means not only that of the environment, but also the first argument of `\multicolumn`.

\hdashline
\cdashline

It is also simple to draw a horizontal dash-line. Use `\hdashline` and `\cdashline` as the counterparts of `\hline` and `\cline`.

For example;

```
\begin{tabular}{|l::c:r|}\hline
A&B&C\\\hdashline
AAA&BBB&CCC\\\cdashline{1-2}
\multicolumn{2}{|l:}{AB}&C\\\hdashline\hdashline
\end{tabular}
```

will produce the following result.

A	B	C
AAA	BBB	CCC
AB		C

Note that the intersections of leftmost/rightmost vertical lines and horizontal dash-lines are little bit different from those produced by ordinary `array/tabular`. That is, with very careful examination you will find that vertical lines of ordinary ones are *broken* with small white specks at intersections, while in the example above they have no specks. In addition, the four corners of outermost rectangular also have specks in ordinary ones, while those in the example above have perfect contacts of L-shape[4].

\firsthdashline
\lasthdashline

If you use array, the dashed version of `\firsthline` and `\lasthline` named `\firsthdashline` and `\lasthdashline` are available.

2.3 Style Parameters

\dashlinedash
\dashlinegap

You have two style parameters to control the shape of dash-lines: `\dashlinedash` is for the length of each dash segment in a dash line; `\dashlinegap` controls the amount of each gap between dash segments. Both parameters have a common default value, 4 pt.

[4] The top-left/right corners had specks before v1.73, the fix in which made the topmost dash segment of a vertical dash-line a little bit shorter.

2.4 Fine Tuning

; Although you can control the shape of dash-lines in an `array`/`tabular` environment as described in §2.3, you might want to draw a dash-line of a shape different from others. To specify the shape of a vertical dash-line explicitly, you may use;

`;{`⟨*dash*⟩`/`⟨*gap*⟩`}`

instead of ordinary '`:`' and will have a dash-line with dash segments of ⟨*dash*⟩ long separated by spaces of ⟨*gap*⟩.

`\hdashline` `\cdashline` As for horizontal dash-lines, explicit shape specifications may be given through optional arguments of `\hdashline` and `\cdashline` as follows.

`\hdashline[`⟨*dash*⟩`/`⟨*gap*⟩`]`
`\cdashline{`⟨*col1*⟩`-`⟨*col2*⟩`}[`⟨*dash*⟩`/`⟨*gap*⟩`]`

For example;

```
\begin{tabular}{|l::c;{2pt/2pt}r|}\hline
A&B&C\\\hdashline[1pt/1pt]
AAA&BBB&CCC\\\cdashline{1-2}[.4pt/1pt]
\multicolumn{2}{|l;{2pt/2pt}}{AB}&C\\\hdashline\hdashline
\end{tabular}
```

will produce the following result.

A	B	C
AAA	BBB	CCC
AB		C

`\ADLnullwide` `\ADLsomewide` The vertical solid and dashed lines are drawn as if their width is zero, as standard LaTeX's `array` and `tabular` do, if you don't use array package. Otherwise, they have *real* width of `\arrayrulewidth` as the authors of array prefers. However, you may explicitly tell arydshln to follow your own preference by `\ADLnullwide` if you love LaTeX standard, or `\ADLsomewide` if you second the preference of array authors.

2.5 Finer Tuning

To draw dash-lines, we use a powerful primitive of TeX called `\xleaders`. It replicates a segment that consist of a dash and gap so that a dash-line has as many segments as possible and distributes *remainder* space to make the spaces between adjacent dash segments (almost) equal to each other. Therefore, you will have dash-lines with consistent steps of gaps and spaces the lines in Figure 1(1) are.

However, because of a bug (or buggy feature) of `\xleaders`, there *had been* a small possibility that a dash segment near the right/bottom end drops, until it was fixed in the version of 3.141592[5]. Though the fix ultimately made any effort to cope with the problem unnecessary, the pacakge still gives you alternative *drawing modes* which you may specify by `\ADLdrawingmode{`⟨*m*⟩`}` as follows.

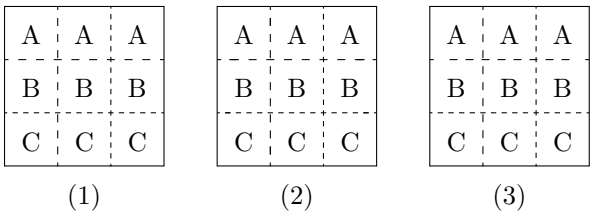

Figure 1: Drawing mode controlled by `\ADLdrawingmode`

`\ADLdrawingmode`

- $m = 1$
  As shown in Figure 1(1), it gives most beautiful result by `\xleaders`[6]. This is default.
- $m = 2$
  As shown in (2) of the figure, beautiful if dash-lines are not so sparse as right/lower lines, but dash segments near the both ends may be a little bit too long as left/upper lines, because in this mode the second first/last segments are drawn by a special mechanism.
- $m = 3$
  As shown in (3) of the figure, beautiful if dash-lines are not so sparse as right/lower lines, but gaps near the both ends may be considerably too large as left/upper lines, because in this mode the lines are drawn by `\cleaders`.

It is strongly recommended to use default mode 1 unless you want to have some special effect.

2.6 Performance Tuning

Since drawing dash-lines is a hard job, you have to be patient with the fact that the performance of typesetting `array`/`tabular` with dash-lines is poorer than that of ordinary ones. In fact, according to author's small performance evaluation with a `tabular` having nine vertical and ten horizontal dash-lines, typesetting the `tabular` is approximately ten times as slow as its ordinary counterpart with solid lines.

However, this is not a really bad news, unfortunately. The real one is that loading `arydshln` makes typesetting `array`/`tabular` slower even if they only have solid lines which the package treats as special ones of dash-lines. The evaluation result shows the degradation factor is about nine. Therefore, if your document has many `array`/`tabular` with solid lines, LaTeX will run slowly even with quite few (or no) `array`/`tabular` with dash-lines,

`\ADLinactivate`

To cope with this problem, you may inactivate dash-line functions by the command `\ADLinactivate` that replaces dash-lines with solid lines drawn by a faster (i.e. ordinary) mechanism. Although the inactivation does not completely solve the performance problem, the degradation factor will become much smaller and acceptable, approximately 1.5 in

[5] By pointing out this problem, the author got a check of $327.68 plus a significantly large amount of interest from DEK. Wow!!

[6] Until the fix of `\xleaders`, the second bottom/rightmost segments of right/lower lines were dropped.

the author's evaluation. For example, the draft version of your document will have the command in its preamble, which you will remove from your final version.

\ADLactivate Alternatively, you may do \ADLinactivate in the preamble, switch on by \ADLactivate before you really need dash-lines, and switch off again afterword. A wiser way could be surrounding array/tabular by \begin{ADLactivate} and \end{ADLactivate}.

Array Tabular If you feel it tiresome to type the long command/environment name for the activation, you may use Array and Tabular(*) environment in which dash-line functions are always active. Note that, however, since these environment names are too natural to keep them from being used by authors of other packages or yourself, name conflict could occur. If Array and/or Tabular have already been defined when arydshln is loaded, you will get a warning to show you have to define new environments, say dlarray and dltabular, as follows.

```
\newenvironment{dlarray}{\ADLactivate\begin{array}}%
                        {\end{array}}
\newenvironment{dltabular}{\ADLactivate\begin{tabular}}%
                          {\end{tabular}}
\newenvironment{dltabular*}{\ADLactivate\begin{tabular*}}%
                           {\end{tabular*}}
```

\ADLnoshorthanded On the other hand, if they are defined after arydshln is loaded, their definitions are silently replaced or LaTeX complains of multiple definitions. The error in the latter case will be avoided by putting \ADLnoshorthanded just after \usepackage{arydshln}.

2.7 Compatibility with Other Packages

Users of array package may use all of newly introduced preamble characters, such as '>', '<', 'm', 'b', and all the commands such as \extrarowheight, \firsthilne and \lasthline. The preamble characters given by arydshln may be included in the second argument of \newcolumntype.

Also users of colortab package may use \LCC/\ECC construct to color columns. A horizontal solid/dash line may be colored by, e.g. \NAC\hdashline\ENAC. The pair of \AC and \EAC may be used to color everything between them *but*, unfortunately, vertical lines are not. There are no ways to color vertical lines in a table having dash lines. You may color vertical lines of a ordinary table inactivating dash line functions by \ADLinactivate.

Another (and more convenient) table coloring tool colortbl may be also used simply by loading it before arydshln. Not only the painting commands \rowcolor, \columncolor and \cellcolor work well, but both solid and dash lines are also colored by the command \arrayrulecolor of colortbl[7]. One caution is that \arrayrulecolor defines the color of the dash-part of dash lines and thus gap-part has no color (i.e. color of the paper on which the line drawn). Therefore, if you have a tabular like;

```
\begin{tabular}{|>{\columncolor{red}}l:>{\colomncolor{green}}r|}
...
\end{tabular}
```

[7]The colortbl manual says \arrayrulecolor and \doublerulesepcolor may be in >{...} in a preamble but they cause an error with the original implementation. This bug is fixed in arydshln and they are now usable to specify the color of the vertical (dash) lines whose specifications occur after the commands.

you will find the vertical dash line is a sequence of black (or the color of `\arrayrulecolor`) and white segments. This problem is partly solved by declaring `\ADLnullwide`[8] to conjunct the red and blue columns and to draw the dash line on their border.

`\ADLnullwidehline` `\ADLsomewidehline` Unfortunately, however, `\ADLnullwide` does not affect the real width of horizontal (dash) lines and thus you will still see white gaps in `\hdashline` and `\cdashline`. A solution is to put `\ADLnullwidehline` before you start a `array`/`tabular`[9]. With this command, a horizontal (dash) line is drawn adjusting its bottom edge to that of the row above. The command `\ADLsomewidehline` turns the switch to default and the top edge of a horizontal (dash) line will be adjusted to the bottom edge of the row above.

`\dashgapcolor` `\nodashgapcolor` Another method to avoid white gaps is to give a color to gaps by `\dashgapcolor` with arguments same as `\color`. For example;

```
\arrayrulecolor{green}\dashgapcolor[rgb]{1,1,0}
```

makes colorful dash lines with green dashes and yellow gaps. The command can be placed outside of `array`/`tabular` for dash lines in the environment, in the argument of preamble character `>` for vertical dash lines following them, or at the beginning of a row for horizontal dash lines following the command. The commmand `\nodashgapcolor` (no arguments) nullifies the effect of `\dashgapcolor`. Note that `\nodashgapcolor` is different from `\dashgapcolor{white}` because the former makes gaps *transparent* while the later whiten them.

`longtable` `Longtable` Usage of longtable with arydshln is quite simple. Just loading arydshln after longtable is enough to make the `longtable` environment able to draw dash-lines. A shorthand activation of dash-line functions is also available by `Longtable` environment. One caution to longtable users is that the temporary results before the *convergence* of the column widths may be different from those without arydshln. For example, the following is the first pass result of the example shown in Table 3 of the longtable manual.

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

Since `LTchunksize` is one in the example, columns of each row has their own widths and thus has vertical lines drawn at the edges of the columns. On the other hand, you will have the following as the first pass result with arydshln.

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

As you see, the vertical lines are drawn at the column edges of the last row[10] because arydshln draws them when it see the last row. Anyway, you may ignore temporary results and will have a compatible result when the column widths are converged like the following.

[8] Since colortbl automatically loads array, the default is `\ADLsomewide`

[9] This command also makes `\cline` and `\cdashline` visisble even if the row below is painted.

[10] More precisely, drawn according to the column widths established by all the chunks preceding page output.

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

3 Known Problems

There are following known problems.

1. The new preamble specifiers '`:`' and '`;{`⟨*dash*⟩`/`⟨*gap*⟩`}`' cannot be followed or preceded by '`@{`⟨*text*⟩`}`', or you will have an ugly result. More specifically, a specifier to draw a dash-line at the left edge of a column cannot be preceded by '`@{`⟨*text*⟩`}`', while that to draw at the right edge cannot be followed by '`@{`⟨*text*⟩`}`'.

2. If you use array package, the restriction of '`@`' shown above is also applied to '`!`'.

3. In order to make it sure that a dash-line always *touches* its both end, i.e. a dash-line always begins and ends with a dash segment, the amount of a gap will slightly vary depending on the dash-line length.

4. If a dash-line is too short, you will have an ugly result without overfull message. More specifically, in mode 1 or 3, a line will look to protrude beyond its column/row borders if it is shorter than a half of `\dashlinedash`. In mode 2, the minimum length to avoid the protrusion is 1.5 × `\dashlinedash` + `\dashlinegap`.

5. As described in §2.6, the processing speed for `array` and `tabular` environment will become slower even if dash-lines are not included.

6. As described in §2.7, `\AC` and `\EAC` pair of colortab such as `\AC&\EAC` cannot color the vertical line at `&`. Use `\ADLinactivate` if you want to have a ordinary table with colored vertical lines. Note that you may color vertical lines with colortbl package.

7. There should be a number of packages whose own `array`/`tabular` implementations are not compatible with arydshln, though the author has made efforts at the compatibility. One of them is plext package for Japanese typesetting but it has a style file named `plextarydshln.sty` to solve the compatibility issue. So if you use the functionality of arydshln with plext, do `\usepackage{plextarydshln}` instead of `\usepackage{arydshln}`.

Acknowledgments

The author thanks to Monty Hayes who gave the author the opportunity to make this style, and Weimin Zhang and Takahiro Kubota who pointed out bugs in early versions. He also thanks to the following people; Sebastian Rahtz and Graham Williams who kindly invited the style to TEX CTAN and online catalogue compiled by Graham; Peter Ehrbar who showed the style was incompatible with array and kindly accepted the offer to be an alpha-user of v1.4 alone; Zsuzsanna Nagy who reported another incompatibility problem with

colortab; Ralf Heydenreich who reported the bug causing that glues in a column have no effect; Yaxin Liu who reported the incompatibility bug of array and `\ADLinactivate`; Craig Leech who reported the incompatibility problem with longtable, which was also reported by Uwe Jehmlich, Torge Thielemann and Florian Weig, and had waited for two years and a half (!) for the solution; Klaus Dalinghaus who reported yet another incompatibility with colortbl; Morten Høgholm who reported the bug of `m`-type columns of array which had not manifested in five (!!) years since the author realesed the first array-compatible version; Maïeul Rouquette who reported another bug of `m`-type columns of longtable with array which had peacefully hidden in the package for eleven years and a half (!!!) since the author made the bug fix shown above carelessly, yet another bug related to longtable, and most surprisingly a problem on intersections of horizontal and vertical (dash-)lines which has hidden for 23 years (!!!!) since the very first version of the package; and Hironobu Yamashita who pointed out bugs hidden for 19 years (!!!!!) by which delarray did not work, and compatibility problems with array v2.4i and longtable in latex-tools 2019-01-05.

The base implementation of `array` and `tabular` environments, part of which the author gives new definitions referring original ones, are written by Leslie Lamport as a part of LaTeX-2.09 and LaTeX 2_ε (1997/12/01) to which Johannes Braams and other authors also contributed. The author also refers array package (v2.4j) written by Frank Mittelbach and David Carlisle; colortab package (v0.9) written by Timothy van Zandt; and longtable (v4.11) and colortbl (v0.1j) packages written by David Carlisle; to make the style compatible with those packages.

Index

Italicized number refers to the page where the specification of corresponding entry are described.